

EXHIBIT 1

FILLED UNDER SEAL

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Exhibit 033-3

Invalidity Claim Chart for U.S. Patent No. 10,779,033 (“the ‘033 patent”)

YouTube Remote and YouTube Leanback (“YouTube Remote”), including the YouTube Remote application and associated servers (e.g., MDx/Lounge server) and screens (“the YT Remote System”) was described in a printed publication, or in public use, on sale, sold, known in this country, or otherwise available to the public before the priority date of the ‘033 patent.

For example, the YouTube Remote was available a first released no later than November 9, 2010. <https://www.eweek.com/it-management/youtube-remote-comes-to-android-market-for-leanback/> (<https://www.eweek.com/it-management/youtube-remote-comes-to-android-market-for-leanback/>) (“Google Nov. 9 launched YouTube Remote to let U.S. users control the YouTube Leanback application from their Android smartphones... Available in the Android Market now”). ~~Features of the YouTube Remote would have been apparent to a person of ordinary skill in the art using the public systems, rendering the systems themselves.~~ Prior to Sonos’s effective priority date (December 30, 2011) and Sonos’s alleged invention date (July 15, 2011), Google conceived of and diligently reduced to practice a “Party Mode” feature for the YouTube Remote. For example, Google conceived of the Party Mode feature by November 2010. See, e.g., GOOG-SONOSNDCA-00075580 at -4274 [referring to launch of Version 1 of the YouTube Remote], -4274 [identifying “party/family mode” as a “plan[] for the future”]; GOOG-SONOS-NDCA-00086353 (YouTube Remote Control – Roadmap referring to “party mode” and implementation of such high priority feature no later than “Q2 2011”). Google thereafter diligently reduced to practice the Party Mode feature and launched it no later than July of 2011 as part of Version 2 of the YouTube Remote.¹ GOOG-SONOSNDCA-00075593 [“Party Mode... Launched in the version 2 of the remote”]; GOOG-SONOS-NDCA-00113784 (showing testing of “party mode” by at least June 2011); GOOG-SONOS-NDCA-00108594 (describing party mode), GOOG-SONOSWDTX-00052947 (same); see also <https://web.archive.org/web/20120323165536/http://www.appbrain.com/app/youtube-remote/com.google.android.ytremote> (change log showing YouTube Remote Version 2.03 was released by July 29, 2011); <https://web.archive.org/web/20110822085859/http://www.appbrain.com:80/app/youtube-remote/com.google.android.ytremote> (referring to “party mode” for Version 2.0.7). Google also diligently released additional updates for the YouTube Remote that contain the party mode feature and are also prior art under at least § 102(a), (b) and (g) ~~prior art~~. See <https://web.archive.org/web/20120323165536/http://www.appbrain.com/app/youtube-remote/com.google.android.ytremote> (see change log describing Version 2.07, 3.0.1 and 3.1.0); see also Dkt. No. 211-14 (January 2012 release of Version 3.1.0).

In its recent August 2, 2022 order granting Google’s Motion for Summary Judgment, the Court construed the term “playback queue” to mean “a list of multimedia content selected for playback.” Dkt. No. 316 at 5. Under the Court’s construction of “playback queue” and Sonos’s interpretation of a “remote” playback queue, the YouTube Remote with Party Mode anticipates or at least renders obvious the asserted claims of the ‘033 patent.

¹ See also Google’s 7-12-2011 capture of YouTube Remote source code (in 2022-03-22 YTRemoteLeanbackAppsServer07122011) including Party Mode feature.

At least the following documents describe the functionality of the YT Remote System:

- [1] <https://youtube.googleblog.com/2010/11/control-youtube-on-desktop-or-tv-with.html> <https://youtube.googleblog.com/2010/11/control-youtube-on-desktop-or-tv-with.html>, By Kuan Yong, Senior Product Manager, Nov.09.2010
- [2] <https://www.youtube.com/watch?v=txIPVu6yngQ> <https://www.youtube.com/watch?v=txIPVu6yngQ>, posted Nov 9, 2010
- [3] "Remote Screen pairing – implementation"
- [4] <https://www.youtube.com/watch?v=EGdsOslqG2s> <https://www.youtube.com/watch?v=EGdsOslqG2s>, Nov 14, 2010
- [5] <https://lifehacker.com/remote-control-youtube-on-your-tv-or-computer-from-your-5685752> <https://lifehacker.com/remote-control-youtube-on-your-tv-or-computer-from-your-5685752>, Nov 9, 2010
- [6] YouTube Lounge Youbiquity Presentation
- [7] MDx protocol as of 2011, as evidenced by MDx Protocol v2 (12/21/11 at 8:06am)
- [8] US 9,490,998
- [9] <https://web.archive.org/web/20111014181427/https://market.android.com/details?id=com.google.android.ytremote>
- [10] <https://palblog.fxpal.com/?p=4953>, Lean back with YouTube and Android by Surendar Chandra, November 11, 2010 (available at <https://web.archive.org/web/20111106221315/https://palblog.fxpal.com/?p=4953>)
- [11] Engadget Article, YouTube Remote app released, controls Leanback on GTV or PC from your Android phone, November 9, 2010
- [12] [GOOG-SONOS-NDCA-00075593](https://www.google.com/patents/US20110108594)
- [13] [GOOG-SONOS-NDCA-00108594](https://www.google.com/patents/US20110108594)

Google also relies on Google source code, both server-side code and device-side code, including any written source code, source code in production, and released source code, including the exemplary code paths and citations referred to below. Google expressly reserves the right to rely on additional source code at a later time.⁺²

⁺² Google has made available for inspection and cited in this [charts chart](#) versions of the source code for the YouTube remote that predate the December 30, 2011 priority date that Sonos identified in its invalidity contentions. ~~Google is also making available for inspection earlier versions,~~ including a July 12 and December 1, 2011 version of the [source code from](#). Because Sonos alleges a July 15, 2011 invention date for the '033 patent, Google cites to the July 12, 2011, ~~including Version 1~~ capture of the [LeanBack code and its corresponding application and server code](#) YouTube Remote source code in this chart. However, ~~that predate Sonos's alleged invention date of July 15, 2011. While~~ Google does not agree that Sonos is entitled to its alleged invention date, ~~to the extent Sonos is entitled to such date Google and~~ may rely upon the same or similar functionality in the ~~earlier~~ [December 1, 2011](#) source code.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Google identifies the authors, designers and implementers of the documents and source code identified and cited herein as prior inventors for purposes of Section 102(g), including but not limited to Ramona Bobohalma.

To the extent publicly available, these documents themselves are also each individually prior art under § 102(a), (b) or (e) and § 103 based on their dates of publication and public availability.

To the extent it is argued the YT Remote System does not disclose any element, that element would be obvious based on the state of the art and/or in combination with one or more of the references noted in Riders I-K.

To avoid duplication and cumulative excerpts, exemplary quotations and citations are provided. The citations to portions of any reference in this chart are exemplary only. Google reserves the right to use the entirety of any reference cited in this chart to show that the asserted claims are anticipated and/or obvious, or to show the state of the art at the relevant time. References to figures should be understood to also refer to any accompanying text. Additional support can be found elsewhere in the prior art reference, and Google expressly reserves the right to rely on such other support and passages at a later time. The use of claim terms in the below chart is based on Sonos' construction of claim terms in its infringement contentions as understood by Google, as well as the plain and ordinary meaning of the claim terms. This chart should not be construed as consenting to or agreeing with Sonos' construction of claim terms. Because discovery is ongoing, Google reserves all rights to amend its invalidity contentions based on new information produced in discovery.

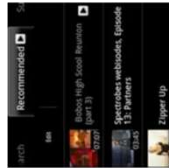
Google expressly reserves the right to supplement its invalidity contentions, including this chart, to demonstrate that the prior art invalidates the claims of the '033 patent.

Claim 1 Portion	Claim 1 Text	YT Remote System
[1Pre]	A computing device comprising: at least one processor; a non-transitory computer-readable medium; and program instructions stored on the non-transitory computer-readable medium that, when executed by the at least one processor, cause the computing device to perform functions comprising:	YT Remote System discloses a computing device (e.g. a phone or computer) having at least one processor, a non-transitory computer readable medium; and program instructions stored on the non-transitory computer-readable medium that, when executed by the at least one processor, cause the computing device to perform the recited functions. <i>See e.g. [1]:</i> "YouTube Remote creates a virtual connection between your phone and YouTube Leanback. To 'pair' your phone with your Leanback screen, simply sign into YouTube Remote on your Android phone, and to YouTube Leanback on your Google TV or computer with the same YouTube account. Just like that, you've connected your powerful multi-touch Android screen with the biggest screen in your home. Once connected, you can use the rich browse and discovery interface on YouTube Remote to find and queue up videos to watch, and send them all to Leanback with a single tap. With

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

YouTube Remote you can play, pause, skip forward and back and even control the sound volume.”

See e.g. [5]



Android: YouTube Remote is a free remote control tool that links the full-screen experience of YouTube Leanback with the convenience of having a touch-screen remote and playlist builder on your Android device.

See also [7] e.g. Sample Session, Remote to Server and Remote to Screen messages

See e.g. [8] at 1:39-50:

In general, this disclosure is directed to techniques for exchanging information between a networked device. Such as a network-enabled television, and web-enabled device, Such as a remote control, via a network service (e.g., a "cloud service"). In an example, the web-enabled device can transmit control information via the network service to the networked device to control playback of media content (e.g., audio and/or video content) on the networked device. In

another example, the networked device can transmit content information via the network service to the web-enabled device. Such as status information concerning the networked device.

See also [8] at 3:14-55:

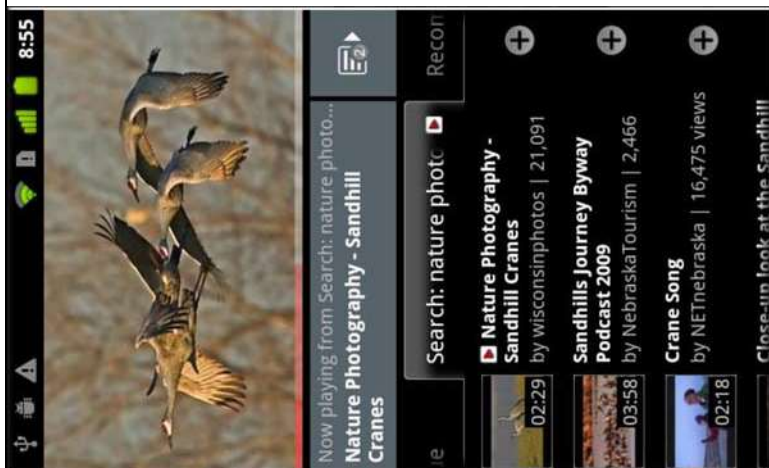
Techniques of this disclosure relate to a network service or "cloud service" that acts as an intermediary between a remote control device and a controlled device. For example, the network service may receive commands from a remote control and transmit the commands to a controlled device. The network service may also receive commands or other information from the controlled device and transmit those commands or other information to the remote control. The remote control may include a remote control application executing on a mobile device. Such as a cellular telephone or a tablet computer. The controlled device may include any Internet-connected device capable of receiving commands, Such as an Internet-connected television, a set top box, a personal video

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

recorder, a gaming console, or other net worked device. In one aspect, the remote control and the controlled device may operate as simple Hypertext Transfer Protocol HTTP clients of the network service. That is, the controlled device does not operate as a server to the remote control. Thus, any HTTP-enabled device may operate as a remote control or as a controlled device. In general, the remote control and the controlled device are configured to both listen for messages from the network service and send messages to the network service. In some examples, the network service controls pairing one or more remote controls and one or more controlled devices, receives information or commands from remote controls and controlled devices, and sends information or commands to remote controls and controlled devices. The network service may direct received information and commands to the appropriate devices based on pairing information maintained by the network service. A remote control may be configured to send a message to a controlled device to perform a task, Such as stopping playback of media content playing on the controlled devices or changing the media content playing on the controlled devices. To accomplish the task, the remote control first sends a message to the network service. The network service then determines the controlled device that is paired with the remote control and forwards the message to the appropriate controlled device. The controlled device receives the message from the network service and performs the task in response to receiving the message.

See e.g. [9]:

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY



See for example source code for the ~~Android application before 12.30.2011, including for example 12.1.2011 capture of the YouTube Remote source code located at google3/java/com/google/android/apps/ytlounge/src/com/google/android/~~

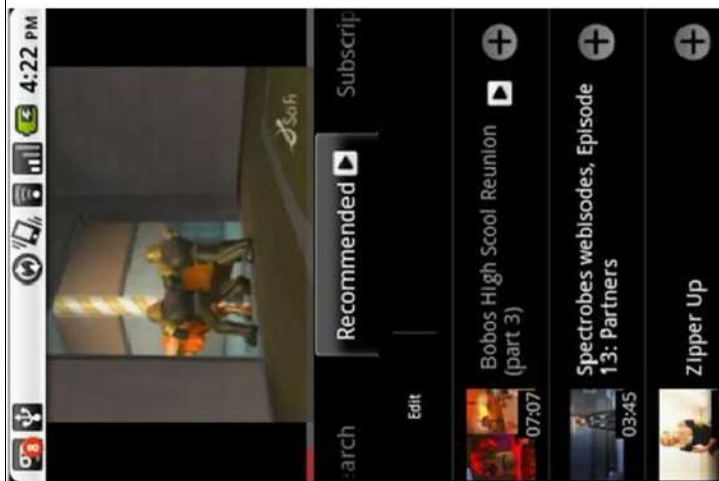
~~To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Riders I. Further discussion of the obviousness of this claim element is provided in Google's Invalidity Contentions Cover Pleading. See also for example the July 12, 2011 capture of the~~

~~YouTube Remote source code located in 2022-03-22_YTTRemoteLeanbackAppsServer07122011.~~

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

[1a]	<p>operating in a first mode in which the computing device is configured for playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service;</p>	<p>YT Remote System discloses the computing device (e.g. a phone or computer) operating in a first mode in which the computing device is configured for playback of a remote playback queue (e.g. a YouTube watch-next queue) provided by a cloud-based computing system (e.g. the MDx server) associated with a cloud-based media service (e.g. a YouTube content server)</p> <p><i>See e.g.</i> [5]</p> <p>YouTube Remote is a simple but effective remote tool for controlling YouTube Leanback from the comfort of your Android device. One of the best features of YouTube Remote is that it isn't just a remote control device for YouTube Leanback, it's also a compact YouTube viewer.</p> <p>You can, for example, preview a video on your Android device before kicking it over to the playlist for your monitor or television. Once you've queued up a video to play on the big screen you can then turn off the remote function and continue to preview and add more videos to the queue.</p>
------	--	---

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY



See e.g. [7], the remote (computing device) has a current playlist, saved in the server (a cloud based computing system associated with a cloud based media service):

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>For a lounge session there is only one playlist being played, the 'Now playing' list. The server keeps track of the playlist</p> <p>Remote to Server messages</p> <p>setPlaylist(videoids, videoid, currentTime)</p> <ul style="list-style-type: none">videoids - comma separated videoid values representing the current playlistcurrentTime - playback position in the video in secondsvideoid - id of the video currently playing, must be part of the playlist <p>The remote informs the server what its current playlist is</p> <p>Sent when the remote connects to a screen</p> <p>If there is no playlist in the session, the playlist sent by the remote will become the current playlist. If there already is one, the playlist will be ignored. The server will also send a request for nowPlaying to the screen</p> <p>addVideo(videoid)</p> <p>insertVideo(videoid)</p> <p>addVideos(videoids)</p> <p>moveVideo(videoid, delta)</p> <p>removeVideo(videoid)</p> <p>clearPlaylist()</p> <p>These messages change the current playlist on the server. If a change occurs, the server will send updates to the remotes (via playlistModified) and to the screen (via updatePlaylist)</p> <p><i>See e.g. [6]</i></p>
--	---

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

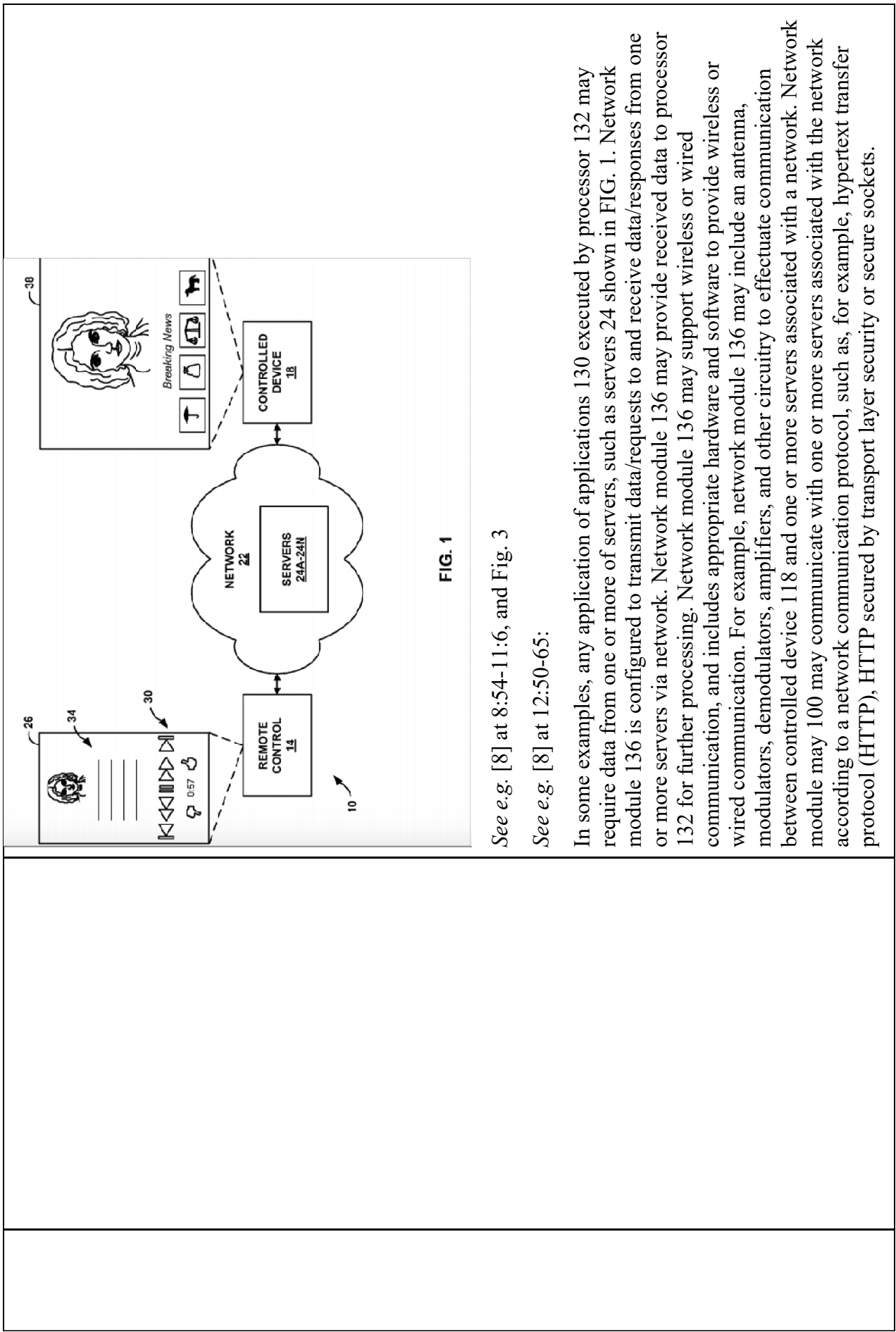
Uniform: seamless transition between phone and big screen

- Android phone magically turns into a remote control for the big screen
- When the user turns off the big screen the viewing experience is transferred to the mobile device
- When the user is back home, the experience is again automatically transferred to the big screen



See e.g. [8] at 4:58-67:

FIG. 1 is a block diagram illustrating an example networked environment 10 with a remote control 14 and controlled device 18, in accordance with one aspect of the present disclosure. According to an aspect of the disclosure, remote control 14 communicates with controlled device 18 via network 22 and servers 24A-24N (collectively “servers 24”) in network 22. As shown in FIG. 1, according to some examples, remote control 14, controlled device 18, and servers 24 may be distinct components (e.g., physically distinct).



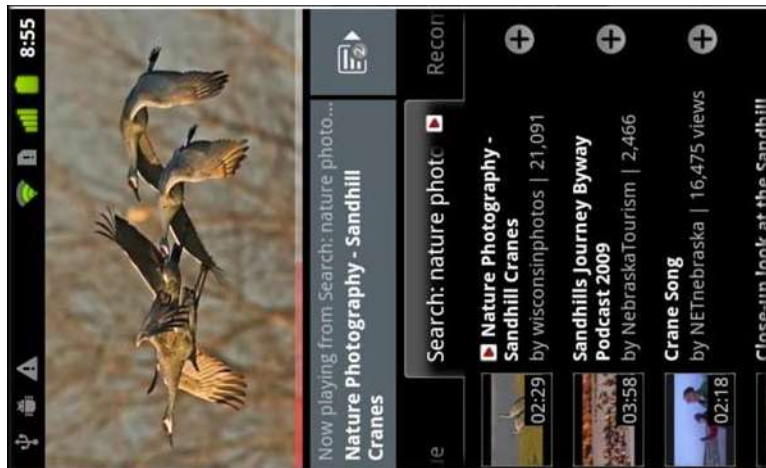
See e.g. [8] at 8:54-11:6, and Fig. 3

See e.g. [8] at 12:50-65:

In some examples, any application of applications 130 executed by processor 132 may require data from one or more of servers, such as servers 24 shown in FIG. 1. Network module 136 is configured to transmit data/requests to and receive data/responses from one or more servers via network. Network module 136 may provide received data to processor 132 for further processing. Network module 136 may support wireless or wired communication, and includes appropriate hardware and software to provide wireless or wired communication. For example, network module 136 may include an antenna, modulators, demodulators, amplifiers, and other circuitry to effectuate communication between controlled device 118 and one or more servers associated with a network. Network module 100 may communicate with one or more servers associated with the network according to a network communication protocol, such as, for example, hypertext transfer protocol (HTTP), HTTP secured by transport layer security or secure sockets.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

See e.g. [9]:



See for example source code for the Android application before ~~12.30.2011~~ 12.1.2011, including for example source code located in subdirectories YTR²³/src/com/google/android/ytremonote; google3/video/youtube/src/web/javascript/library; google3/java/com/google/net/browserchannel/. google3/javascript/closure/net/ and google3/video/youtube/src/web/javascript/library/tv/views/video/.

See also e.g.:

²³ Throughout, "YTR" refers to google3/java/com/google/android/apps/ytlounge/

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

YTR/src/com/google/android/ytremote/backend;

YTR/src/com/google/android/ytremote/backend/station;

YTR/src/com/google/android/ytremote

Further, on August 2, 2022, this Court construed the term “playlist queue” as “a list of multimedia content selected for playlist.” The Court indicated that a “playlist queue” is not limited to a list of multimedia items selected by the user for playlist and need not have multiple media items. At least under the Court’s construction of “playlist queue” and Sonos’s interpretation of a “remote” playlist queue and “cloud-based media service,” this limitation is disclosed or at least obvious based on the YT Remote System’s disclosure of a “party” (or shared) playlist provided by a cloud-based media service. Thus, a phone or tablet running YouTube Remote Version 2.0 or later with “party mode” may be configured for playlist of a remote party queue provided by the Google cloud servers (e.g., MDx/Lounge servers and Bandaid servers).

For instance, a user with a phone or tablet running YouTube Remote Version 2.0 or later with “party mode” may use his or her computing device to create a playlist and may playlist the playlist on his or her phone. The user may further invite one of more guests to the party.⁴ In party mode, the “host” user sends a playlist to a cloud server (e.g., an MDx or Lounge server) along with the list of guests. For instance, in the July 12, 2011 capture of the YouTube Remote source code, the inviteToParty function⁵ retrieves the current media queue⁶ and calls the partyModeManager.initPartyMode.⁷ Thereafter, the host YouTube Remote application goes to party mode.⁸ The initPartyMode function⁹ receives the video queue and guest name(s), and creates the new party queue. The play queue is then set to the

⁴ See, e.g., ytremote/ContactListActivity.java, lines 106 (inviteToParty), 197-208 (onActivityResult), 204 (calls inviteToParty).

⁵ See, e.g., *id.* at lines 389-432.

⁶ See, e.g., *id.* at lines 419-423.

⁷ See, e.g., *id.* at lines 424.

⁸ See, e.g., *id.* at lines 428-430.

⁹ See, e.g., ytremote/backend/RealPartyModeManager.java at lines 189-212.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>party queue¹⁰ and the <code>initPartyMode</code> function then sends a <code>INIT_PARTY_MODE</code> message to lounge server¹¹. This message contains the video queue¹², the current video¹³, the current time¹⁴, and guest name(s)¹⁵.</p> <p>The lounge server receives the playlist sent by the host user which it stores as a party playlist (or party queue). For instance, the <code>INIT_PARTY_MODE</code> message sent by the host user is received by the YT Lounge server, and its <code>handleMessage</code> function¹⁶. The <code>initPartyMode</code> function¹⁷ sets the lounge session queue (the party queue) to be the queue received from the host¹⁸. The <code>sendInvitations</code> function¹⁹ sends a <code>PARTY_INVITE</code> message to selected guest users for the party. This message contains the name and type of the party queue²⁰.</p> <p>Thereafter, the guests (should they accept the party invitation) can receive the party queue that is stored in the cloud. For example, each guest user's YTR application receives the <code>PARTY_INVITE</code> message²¹ which causes the YTR application to issue a <code>PARTY_INVITE</code> intent. The intent contains the name of the host, the name of the shared</p>
--	--

¹⁰ See, e.g., *id.* at line 196.

¹¹ See, e.g., *id.* at line 211.

¹² See, e.g., *id.* at line 205.

¹³ See, e.g., *id.* at line 207.

¹⁴ See, e.g., *id.* at line 208.

¹⁵ See, e.g., *id.* at line 210.

¹⁶ See `youtube/lounge/browserchannel/RealLoungeSessionManager.java` at line 326.

¹⁷ See, e.g., *id.* at line 776.

¹⁸ See, e.g., *id.* at line 779-785.

¹⁹ See, e.g., *id.* at line 946.

²⁰ See, e.g., *id.* at lines 795-796; see *also* `model\PartyQueue.java`; `backend\RemoteQueueManager.java`

²¹ See, e.g., `ytremote/backend/CloudServiceMessageListener.java` at line 137.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

		<p>party queue, and other information.²² This intent is serviced by BaseActivity.java (and a few other components) and calls onPartyInvite with the received host and station name.²³ The onPartyInvite function²⁴ calls showInvite²⁵. The showInvite function²⁶ creates a dialog, and allows the invited guest user to accept or reject party invite²⁷. If the user accepts²⁸ the partyModeManager.joinParty function is called. The play queue is set as a new video play state (from remote queue), and the YTRemote application connects to the host's lounge.²⁹ Upon connecting to the Lounge server, the addDevice function is called³⁰ and causes the Lounge Server to send the cloud hosted party queue to the device³¹. The sendPartyQueueToDevice³² calls sendPartyQueueToRemote (for the new remote YTRemote application that has just joined). The sendPartyQueueToRemote³³ sends a PARTY_PLAYLIST_MODIFIED message to the remote. This message contains the shared party queue³⁴. The guest YTRemote application receives the PARTY_PLAYLIST_MODIFIED message³⁵ with the shared party queue.</p>
--	--	---

²² See, e.g., *id.* at lines 145-158.

²³ See, e.g., BaseActivity.java at lines 104, 125.

²⁴ See, e.g., *id.* at line 412.

²⁵ See, e.g., *id.* at line 413.

²⁶ See, e.g., *id.* at line 460.

²⁷ See, e.g., *id.* at lines 463-479.

²⁸ See, e.g., *id.* at lines 480-491.

²⁹ See, e.g., *id.* at lines 486, 489.

³⁰ See, e.g., RealLoungeSessionManager.java, line 237.

³¹ See, e.g., *id.* at lines 260-264.

³² See, e.g., *id.* at line 1065.

³³ See, e.g., *id.* at line 1074.

³⁴ See, e.g., *id.* at line 1082.

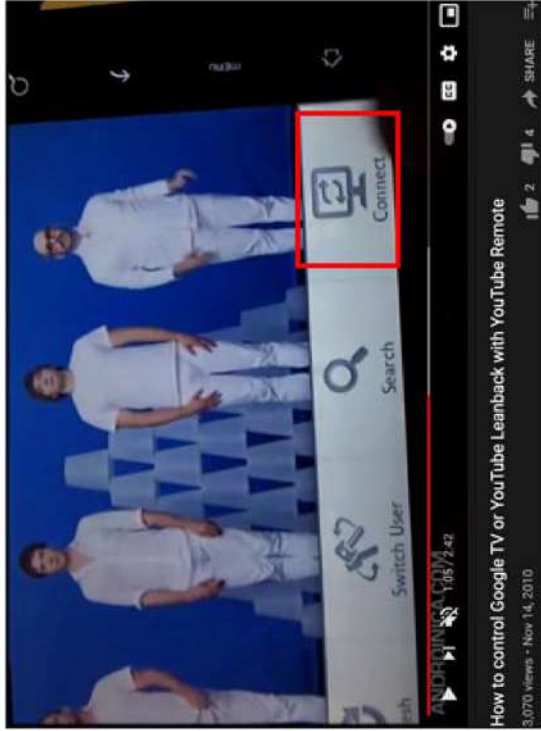
³⁵ See, e.g., CloudServerMessageListener.java, line 96.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

<p>The host and guest users can also continue to edit and manage the remote party queue:</p> <pre> ** * Content service implementation for party mode. The definitive version of the * playlist lives on the server, and multiple remote controls can change it at * the same time. * * @author bobohalma@google.com (Ramona Bobohalma) */ vremote\backend\SharedPlaylistContentService.java </pre> <p>The YouTube Remote system, therefore, discloses, or at least renders obvious, operating in a first mode in which the computing device is configured for playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service (e.g., playback of a party playlist that may be stored in the cloud).</p> <p><i>See, e.g., [12]:</i></p> <div data-bbox="812 128 1274 1323"> <h3>Party Mode</h3> <p>Party mode allows multiple users to connect to the same screen, queue up and watch videos together. All members of the party can add/reorder/delete videos to the same 'Shared Queue' of videos, which is being played on the Leanback screen. It basically solves this problem: http://xkcd.com/920/.</p> <p>I've designed and implemented the shared queue management(both android app and server) cl/18357652 cl/18343670</p> <p>Redesigned the party mode initialization to be invitation based: cl/18677970, cl/18638464</p> <p>Launched in the version 2 of the remote.</p> </div>	<p>To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Rider I.</p>
--	---

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>[1b] while operating in the first mode, displaying a representation of one or more playback devices in a media playback system that are each i) communicatively coupled to the computing device over a data network and ii) available to accept playback responsibility for the remote playback queue;</p>	<p>Further discussion of the obviousness of this claim element is provided in Google's Invalidity Contentions Cover Pleading.</p> <p>The YT Remote System discloses while operating in the first mode, displaying a representation of one or more playback devices in a media playback system that are each i) communicatively coupled to the computing device over a data network and ii) available to accept playback responsibility for the remote playback queue.</p> <p>See e.g. [3]</p> <div data-bbox="526 407 935 1306" style="border: 1px solid blue; padding: 10px;"> <p>This document describes the implementation details of pairing a mobile device (phone or tablet) with a controlled YouTube Leanback screen. See this document for an overview of what we are doing from a product perspective.</p> <p>Terminology</p> <ul style="list-style-type: none"> • pairing code: temporary code used to pair the screen with a mobile device, either manually typed in by the user or scanned from a QR code or slurped via NFC • screen id: unique identifier for the screen; never displayed to the user, only sent via https; this number is large enough to be resistant to collision attacks; this is the permanent code that is used by the screen and mobile device to find each other in the cloud • lounge token: authentication token constructed based on screen id, used to identify the screen to connect to; gives the requester access to the lounge server session; uses a much larger space than the 64 bit screen id, thus making it invulnerable to collision attacks </div> <p>See e.g. [5]:</p> <p>To use YouTube Remote you'll need a YouTube account. Your YouTube login credentials are the glue that binds the Android remote to what's happening on YouTube Leanback. YouTube Remote is a free application, you can download it by scanning the QR code at right or searching for "YouTube Remote" in the Android Market.</p> <p>The YT remote identifies devices connected to the same LAN and connected to the same account. See e.g. [4]:</p>
--	---	---



See e.g. [7]:

Server to Remote messages

playlistModified(videoId, videoId)

! videoId - comma separated video id values representing the current playlist
! videoId - id of the video currently playing, must be part of the playlist
Whenever the current playlist is modified (either by a remote or from the screen) the server sends updates to all remotes in the session.

oungeScreenConnected()

The server informs the remote that there is at least one screen connected in the session

oungeScreenDisconnected(video_id, current_time)

! video_id - the encrypted video id of the currently playing video
! current_time - playback position in the video

The server informs the remote that there is no screen connected in the session
Optionally, if there was a screen before, the server sends info about what the screen was playing when it was disconnected.

See e.g. [8] at 4:21-57:

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>Remote controls and controlled devices may be paired using any one of several different techniques. As one example, a user may maintain a user account using the network service, and the remote controls and controlled devices may be associated with the user account. For example, upon connecting to a network service, the remote controls and controlled devices may notify the network service that the remote controls and controlled devices are connected to the network. The network service may, in some examples, determine whether the remote controls and controlled devices are authorized to be associated with the user account. If authorized, the network service initiates a session and assigns the remote controls and controlled devices unique identification numbers. The network service uses the unique identification numbers for pairing during a session. In another example, a user may be presented with a quick response (“QR”) code via the controlled device that the user scans with the remote control (e.g., using a camera of the remote control). The QR code identifies a user account or previously initiated session maintained by the network Service. Upon Scanning the QR code, the remote control may send a message to the network service indicating that the network service should assign a unique identification number to the remote control and pair the remote control with the user account or session identified by the QR code. In this manner, one or more remote controls may control one or more controlled devices via the network service.</p> <p>Using the network service to transmit and receive messages between a remote control and a controlled device may enable non-traditional devices having rich input and display capabilities to act as a remote control. In addition, by using the network service as an intermediary, the remote control and the controlled device, in various instances, may not need to be connected to the same local area network, nor in physical proximity to each other. The network service may also enable pairing of a nearly limitless number of remote controls and controlled devices.</p> <p><u>See also [8] at 10:63-11:6; U.S. Patent No. 10,469,894 (continuation of ’998 patent), Claim 1</u></p> <p><i>See e.g. [10]:</i></p> <p>“The system even works when the user logs into multiple Leanback browsers; remote control operations are seamlessly sent to all browsers.”</p> <p><i>See e.g. [11]</i></p>
--	--

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Update: Once we got everything rolling, we were able to get a better impression of the app. While it was a bit slow to open on our Galaxy S phone, once it is up, it worked smoothly, scrolling side to side through various queues of types of content and our favorites list. While the task of pulling up Leanback in a browser window or even on a Google TV device makes it ill-suited for viewing just one video at a time, where it excels is building a up a queue of videos and sending them over all at once. It will work on multiple screens at the same time as well, but there's no AirPlay-style syncing to be had, if one of them starts to slow down or buffer it will simply continue lagging behind, and without any volume controls or ability to reach other functions, you'll still need to keep other remotes handy.

See for example source code for the Android application before ~~12.30.2011~~ 12.1.2011, including for example source code located in subdirectories

YTR/src/com/google/android/ytremote/adapt, YTTVF³³⁶/youtube/

YTTVF/youtube/tv/services, YTTVF/youtube/players/ YTTVF/youtube/tv/components,

YTTVF/net/browserchannel/, YTR/src/com/google/android/ytremote/backend/, and

google3/video/youtube/src/web/javascript/library/tv/

google3/video/youtube/src/web/javascript/library/www/remote/

google3/java/com/google/net/browserchannel/, google3/javascript/closure/net/

See also e.g.: YTR/src/com/google/android/ytremote/adapt:

YTR/src/com/google/android/ytremote; YTTVF/youtube/tv/services;

YTR/src/com/google/android/ytremote/backend/station.

Further, on August 2, 2022, this Court construed the term “playback queue” as “a list of multimedia content selected for playback.” The Court indicated that a “playback queue” is not limited to a list of multimedia items selected by the user for playback and need not have multiple media items. At least under the Court’s construction of “playback queue” and Sonos’s interpretation of a “remote” playback queue, the YT Remote System’s shared party playlist (or party queue) is a “remote playback queue.” Thus, a phone or tablet running

³³⁶ Throughout, “YTTVF” refers to YTTVFlashLite12282011/google3/flash/actionsript/com/google/

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

YouTube Remote Version 2.0 or later with “party mode” also satisfies this limitation under the Court’s construction and Sonos’ interpretation of a “remote” playback queue.

For instance, the Version 2.0 of the YouTube Remote, like early versions, included a “connect” button that is a representation of one or more playback devices in a media playback system that are each i) communicatively coupled to the computing device over a data network and ii) available to accept playback responsibility for the remote playback queue. See Bhattacharjee Decl., ¶¶ 136, 142-143. Thus, the YouTube Remote prior art discloses or renders obvious this limitation by, for instance, displaying the “connect” button for transfer.

To the extent it is argued that the “connect” button is not the claimed “representation of one or more playback devices in a media playback system” and that a “device-picker” is required, this limitation is anticipated and obvious based on the device-picker in Google’s December 1, 2011 source code that was released no later than January of 2012 in Version 3 of the YouTube Remote. Bhattacharjee Decl., ¶170. It is also obvious in view of the YouTube Remote Patent’s disclosure of a device-picker, Google’s Tungsten/Nexus Q device with device-picker, Apple Airplay, Sonos’s own prior art, and/or the Al-Shayk patent, as the Court found in its August 2, 2022 Order. Bhattacharjee Decl., ¶¶ 27-34, 165-174; Dkt. No. 316 at 14-17.

See, e.g. [13]:

Temporary pairing (party mode)

User opens the screen management dialog on their mobile device and click ‘share screen’. This will make the screen automatically detectable via the network (using UDP), so all the friends will see the screen popping up in their screen list. We can optionally protect this sharing with a 2 digit pin, but I personally don’t think it’s necessary.

In addition, the phone will display a *pairing code* that can be entered manually, should some devices in the party not be on the same network (e.g. 3G).

To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with prior art including the

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

[1c]	<p>while displaying the representation of the one or more playback devices, receiving user input indicating a selection of at least one given playback device from the one or more playback devices;</p>	<p>references cited in Rider J. Further discussion of the obviousness of this claim element is provided in Google’s Invalidity Contentions Cover Pleading.</p> <p>The YT Remote System discloses while displaying the representation of the one or more playback devices, receiving user input indicating a selection of at least one given playback device from the one or more playback devices;</p> <p>See e.g.: [4], selection of the “Connect” icon by the user</p>  <p>See e.g. [8] at 4:21-57:</p> <p>Remote controls and controlled devices may be paired using any one of several different techniques. As one example, a user may maintain a user account using the network service, and the remote controls and controlled devices may be associated with the user account. For example, upon connecting to a network service, the remote controls and controlled devices may notify the network service that the remote controls and controlled devices are connected to the network. The network service may, in some examples, determine whether the remote controls and controlled devices are authorized to be associated with the user account. If authorized, the network service initiates a session and assigns the remote</p>
------	--	--

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

controls and controlled devices unique identification numbers. The network service uses the unique identification numbers for pairing during a session. In another example, a user may be presented with a quick response (“QR”) code via the controlled device that the user scans with the remote control (e.g., using a camera of the remote control). The QR code identifies a user account or previously initiated session maintained by the network Service. Upon Scanning the QR code, the remote control may send a message to the network service indicating that the network service should assign a unique identification number to the remote control and pair the remote control with the user account or session identified by the QR code. In this manner, one or more remote controls may control one or more controlled devices via the network service. Using the network service to transmit and receive messages between a remote control and a controlled device may enable non-traditional devices having rich input and display capabilities to act as a remote control. In addition, by using the network service as an intermediary, the remote control and the controlled device, in various instances, may not need to be connected to the same local area network, nor in physical proximity to each other. The network service may also enable pairing of a nearly limitless number of remote controls and controlled devices.

See also [8] e.g. at 8:1-59 (Pairing of remote controls with controlled device), [10:63-11:6; U.S. Patent No. 10,469,894 \(continuation of ‘998 patent\), Claim 1](#)

See e.g. [10]

“The system even works when the user logs into multiple Leanback browsers; remote control operations are seamlessly sent to all browsers.”

See e.g. [11]

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Update: Once we got everything rolling, we were able to get a better impression of the app. While it was a bit slow to open on our Galaxy S phone, once it is up, it worked smoothly, scrolling side to side through various queues of types of content and our favorites list. While the task of pulling up Leanback in a browser window or even on a Google TV device makes it ill-suited for viewing just one video at a time, where it excels is building a up a queue of videos and sending them over all at once. It will work on multiple screens at the same time as well, but there's no Airplay-style syncing to be had, if one of them starts to slow down or buffer it will simply continue lagging behind, and without any volume controls or ability to reach other functions, you'll still need to keep other remotes handy.

See for example source code for the Android application before ~~12.30.2011~~ 12.1.2011, including for example source code located in subdirectories YTR/src/com/google/android/ytremote/, and google3/video/youtube/src/web/javascript/library/tv/

See also e.g.:

YTR/src/com/google/android/ytremote/adaptter; YTR/src/com/google/android/ytremote.


To the extent it is argued that the “connect” button is not the claimed “representation of one or more playback devices in a media playback system” and that a “device-picker” is required, this limitation is anticipated or at least obvious based on the device-picker in Google’s December 1, 2011 source code that was released no later than January of 2012 in Version 3 of the YouTube Remote. Bhatacharjee Decl., ¶170. It is also obvious in view of the YouTube Remote Patent’s disclosure of a device-picker, Google’s Tungsten/Nexus Q device with device-picker, Apple Airplay, Sonos’s own prior art, and/or the Al-Shayk patent, as the Court found in its August 2, 2022 Order. Bhatacharjee Decl., ¶¶ 27-34, 165-174; Dkt. No. 316 at 14-17. See also Bhatacharjee Decl., ¶143 (discussing connect button and video fling).³⁷

³⁷

2022-03-

22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/ControlsHelper.java at line 267, 349

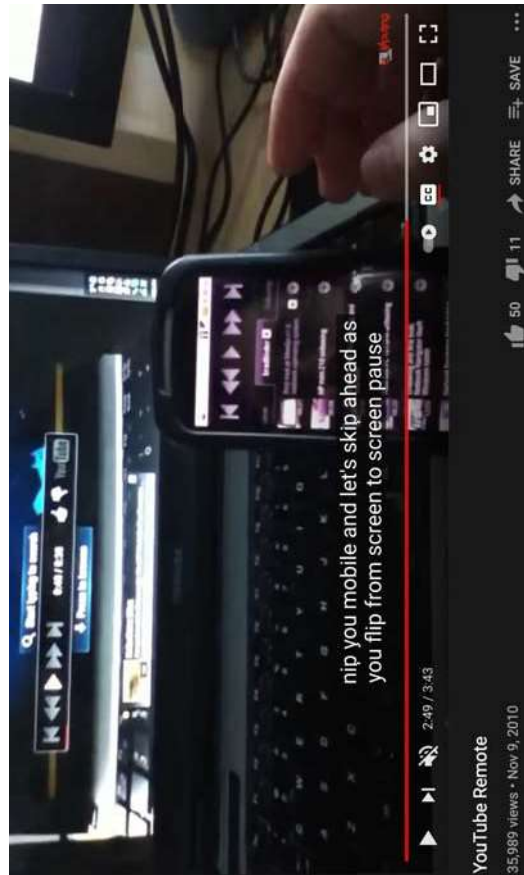
HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

		<p>To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with prior art including the references cited in Rider J. Further discussion of the obviousness of this claim element is provided in Google’s Invalidity Contentions Cover Pleading.</p>
<p>[1d]</p>	<p>based on receiving the user input, transmitting an instruction for the at least one given playback device to take over responsibility for playback of the remote playback queue from the computing device,</p>	<p>The YT Remote System discloses based on receiving the user input, transmitting an instruction for the at least one given playback device to take over responsibility for playback of the remote playback queue from the computing device; <i>See e.g.</i>: [4], based on the selection of the “Connect” icon by the user, the leanback screen takes over responsibility for playback of the remote playback queue from the computing device:</p>  <p>The screenshot shows a YouTube video player interface. At the top, the video title is 'OK Go - White Knuckles - Official Video' by 'OkGo' with 740,129 views. A red box highlights a text overlay that says 'Connected to Leanback screen' with a monitor icon. The video content shows three people in white suits on a stage. The YouTube interface includes a progress bar at the bottom, a timestamp of 1:13 / 2:42, and a title 'How to control Google TV or YouTube Leanback with YouTube Remote' with 3,069 views from Nov 14, 2010. Interaction icons for likes, shares, and saves are visible on the right.</p>

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY



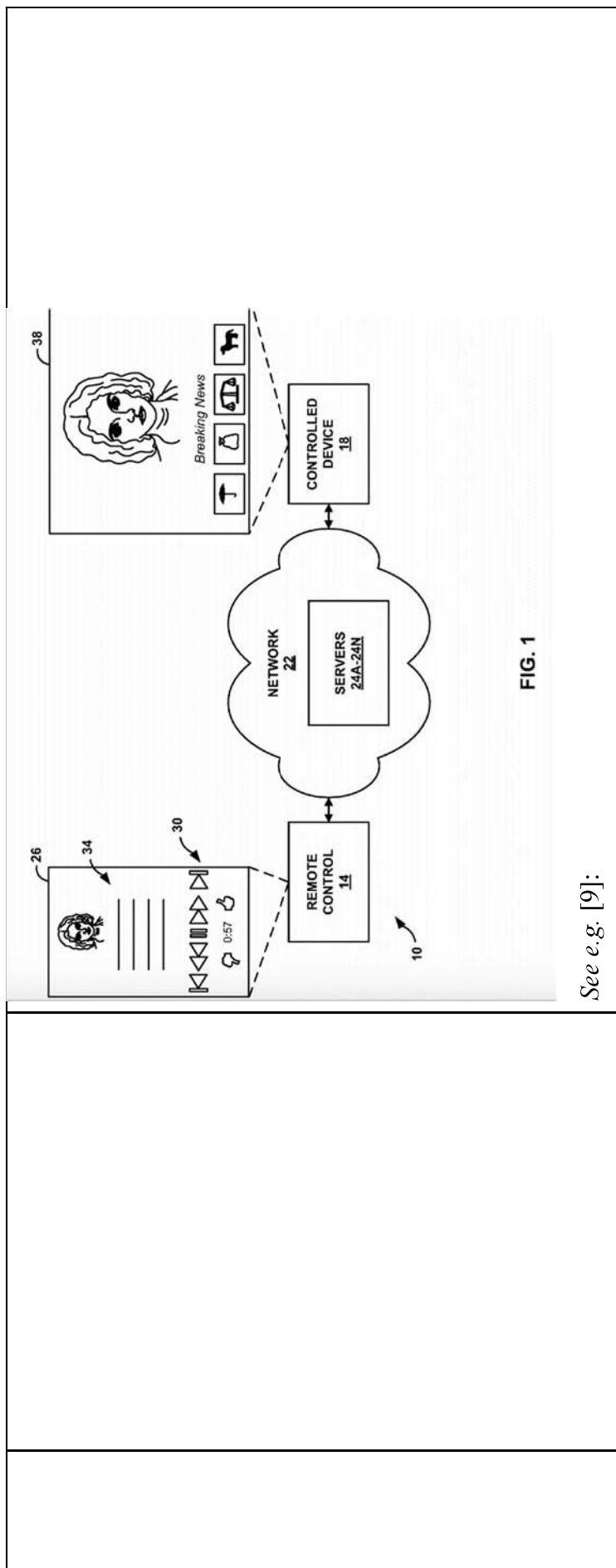
See also e.g. [2]:



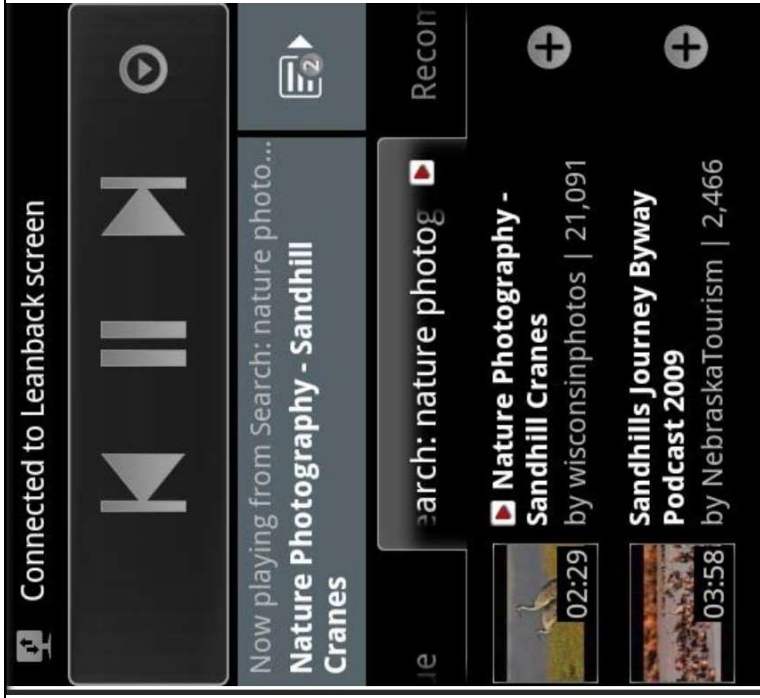
See e.g. [7]:

	<p>For a lounge session there is only one playlist being played, the "Now playing" list. The server keeps track of the playlist.</p> <p>Remote to Server messages</p> <p>setPlaylist(videoId, videoId, currentTime)</p> <p>videoId - comma separated video id values representing the current playlist currentTime - Playback position in the video in seconds videoId - id of the video currently playing, must be part of the playlist</p> <p>The remote informs the server what its current playlist is.</p> <p>Sent when the remote connects to a screen. If there is no playlist in the session, the playlist sent by the remote will become the current playlist. If there already is one, the playlist will be ignored. The server will also send a request for nowPlaying to the screen.</p> <p>addVideo(videoId) insertVideo(videoId) addVideos(videoIds) moveVideo(videoId, delta) removeVideo(videoId) clearPlay() {}</p> <p>These messages change the current playlist on the server. If a change occurs, the server will send updates to the remotes (via playlistModified) and to the screen (via updatePlaylist).</p> <p>See e.g. [8] at 4:58-67:</p> <p>FIG. 1 is a block diagram illustrating an example networked environment 10 with a remote control 14 and controlled device 18, in accordance with one aspect of the present disclosure. According to an aspect of the disclosure, remote control 14 communicates with controlled device 18 via network 22 and servers 24A-24N (collectively "servers 24") in network 22. As shown in FIG. 1, according to some examples, remote control 14, controlled device 18, and servers 24 may be distinct components (e.g., physically distinct).</p>
--	---

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY



HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

		<p>See [redacted] for example source code for the Android application before 12.30.2011 12.1.2011, including for example source code located at YTR/src/com/google/android/ytre mote/. See also for example source code located in subdirectories YTR/res/values, YTL⁴³⁸/browserchannel, YTTVF/google/youtube/, YTTVF/net/browserchannel, and google3/video/youtube/src/web/javascript/library/www/remote/, google3/java/com/google/net/browserchannel/, google3/javascript/closure/net/.</p> <p>See also e.g.: YTR/res/values; YTR/src/com/google/android/ytre mote/; YTL/browserchannel.</p>
--	--	--

⁴³⁸ Throughout, “YTL” refers to <google3/java/com/google/youtube/lounge/>

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Further, on August 2, 2022, this Court construed the term “playback queue” as “a list of multimedia content selected for playback.” The Court indicated that a “playback queue” is not limited to a list of multimedia items selected by the user for playback and need not have multiple media items. At least under the Court’s construction of “playback queue” and Sonos’s interpretation of a “remote” playback queue, the YT Remote System’s shared “party queue” is a “remote playback queue.” Thus, a phone or tablet running YouTube Remote Version 2.0 or later with “party mode” also discloses and renders obvious this limitation under the Court’s construction.

For example, the YouTube Remote discloses and renders obvious that upon a user input the YouTube Remote application may connect to the host’s lounge³⁹ and send a SET PARTY PLAYLIST message⁴⁰ that includes the party queue⁴¹ to the Lounge server. This message causes the Lounge server to call the sendPartyQueueToScreen⁴². This function sends a SET PLAYLIST and a SET VIDEO message from the Lounge server to the screen. The SET PLAYLIST message sent from the Lounge server to the screen contains the party queue. See *also* Bhattacharjee Decl., ¶143 (discussing connect button and video fling).

See, e.g., [12]:

39

See, e.g., *id.* at ., 2021-11-30 YTRemoteLeanbackAppsServer-

07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/ControlsHelper.java at 266, 277

40

See, e.g., *id.* at line 365

41

See, e.g., *id.* at line 377-381

42



See, e.g., 2021-11-30 YTRemoteLeanbackAppsServer-

07122011/google3/java/com/google/youtube/lounge/browserchannel/RealLoungeSessionManager.java at 1088

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

		<p>Party Mode</p> <p>Party mode allows multiple users to connect to the same screen, queue up and watch videos together. All members of the party can add/reorder/delete videos to the same 'Shared Queue' of videos, which is being played on the Leanback screen. It basically solves this problem: http://xkcd.com/920/.</p> <p>I've designed and implemented the shared queue management(both android app and server) cl/18357652 cl/18343670</p> <p>Redesigned the party mode initialization to be invitation based: cl/18677970, cl/18636464</p> <p>Launched in the version 2 of the remote.</p>
		<p>To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Riders I-J. Further discussion of the obviousness of this claim element is provided in Google's Invalidity Contentions Cover Pleading.</p>
[1e]	<p>wherein the instruction configures the at least one given playback device to (i) communicate with the cloud-based computing system in order to obtain data identifying a next one or more media items that are in the remote playback queue, (ii) use the obtained data to retrieve at least one media item in the remote playback queue from the cloud-based media service; and (iii) play back the retrieved at least one media item;</p>	<p>The YT Remote System discloses that the instruction configures the at least one given playback device to (i) communicate with the cloud-based computing system in order to obtain data identifying a next one or more media items that are in the remote playback queue, (ii) use the obtained data to retrieve at least one media item in the remote playback queue from the cloud-based media service; and (iii) play back the retrieved at least one media item;</p> <p><i>See e.g.</i> [4], following the instruction from the control device to the Leanback screen, the Leanback screen communicates with the YT cloud-based computing system in order to obtain data identify a next one or more media items in the remote playback queue, retrieving at least one media item (the OK Go – White Knuckles – Official Video) in the remote playback queue, and playing it back.</p>

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

 <p>OK Go - White Knuckles - Official Video by OkGo 7401299 views</p> <p>Connected to Leanback screen</p> <p>How to control Google TV or YouTube Leanback with YouTube Remote</p> <p>3,069 views · Nov 14, 2010</p>	 <p>One or more media items in the remote playback queue</p> <p>How to control Google TV or YouTube Leanback with YouTube Remote</p> <p>3,069 views · Nov 14, 2010</p>
<p>The user’s YT account includes playlists or channels, with videos automatically added to the user’s feed. The playlist or channel comprise multimedia content added to a queue, which is accessed on the leanback screen.</p> <p>“When you search, your results end up in a channel and as soon as you finish playing one video, youtube leanback automatically plays the rest in sequence” (https://blog-youtube-news-and-events.youtube-leanback-offers-</p>	

~~effortless?m=1https://blog.youtube/news-and-events/youtube-leanback-offers-effortless?m=1~~

“Your feed is personalized to you, based on your youtube preferences ... and once one video ends, the next automatically begins” (~~https://blog.youtube/news-and-events/youtube-leanback-offers-effortless?m=1https://blog.youtube/news-and-events/youtube-leanback-offers-effortless?m=1~~)

See e.g. [7] **Remote to Server, Remote to Screen, Screen to Server and Server to Screen:**

For a lounge session there is only one playlist being played, the ‘Now playing’ list. The server keeps track of the playlist

Remote to Server messages

setPlaylist(videoids, videoid, currentTime

- ! videoids - comma separated video id values representing the current playlist**
- ! currentTime - playback position in the video in seconds**
- ! videoid - id of the video currently playing, must be part of the playlist**

The remote informs the server what its current playlist is

Sent when the remote connects to a screen

If there is no playlist in the session, the playlist sent by the remote will become the current playlist. If there already is one, the playlist will be ignored. The server will also send a request for nowPlaying to the screen

addVideo(videoid)

insertVideo(videoid)

addVideos(videoids

moveVideo(videoid, delta

removeVideo(videoid)

clearPlaylist()

These messages change the current playlist on the server. If a change occurs, the server will send updates to the remotes (via playlistModified) and to the screen (via updatePlaylist

	<div><div><div>Server to Screen messages</div><div><div><div>setPlaylist(videos, currentIndex, currentTime)</div><div><div>videos - comma separated video id values representing the current playlist</div><div>currentIndex - playback position in the video in seconds</div><div>currentTime - index of the video currently playing</div></div></div><div>Sets the playlist from the screen and starts playing the current video from the current time. If current video is not present will play the first video in the list. Should not</div></div></div><div><div><div>UpdatePlaylist(videos)</div><div><div>videos - comma separated video id values representing the current playlist</div></div><div>Updates the current playlist. The currently playing video must be in videos</div></div><div><div><div>getPlaylist()</div><div>The server makes a request to the screen to send its current playlist</div></div><div><div><div>getNowPlaying()</div><div>The server makes a request to the screen to send a now playing message</div></div></div></div></div></div>

Screen to Server messages

nowPlaying(video id, current time)

- video id - the encrypted video id of the currently playing video
- current time - playhead position in the video
- state - the video player state: unstarted (-1), ended (0), playing (1), paused (2), buffering (3), video cued (5)

Learnback informs the server what the currently playing video is. The screen sends this message after any video data change event (such as on next, on previous, or next video auto-plays). The server will forward this message to all remotes in the session. If the video id is not found in the current server playlist, the server will issue a getPlaylists() message to the screen.

onStateChange(state)

- state - the new video player state: unstarted (-1), ended (0), playing (1), paused (2), buffering (3), video cued (5)

Implemented via the our player's JS API, where we receive a onStateChange event we relay that information to the remotes.

confirmPlaylistUpdate(updated)

- updated - true/false: whether the playlist was updated

As a response to updatePlaylist, confirms whether the playlist was updated or not. It could be rejected if the playing video is not part of the playlist being sent.

If the response is false, the server will send a setPlaylist message if the session has a non empty playlist.

nowPlayingPlaylist(video ids, current video)

- video ids - comma separated video id values representing the current playlist
- current video id - the encrypted video id of the currently playing video
- state - the video player state: unstarted (-1), ended (0), playing (1), paused (2), buffering (3), video cued (5)

The server will replace the 'nowPlaying' queue with video ids and send a playlistModified(video ids, current video) message to the remote controls.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Remote to Screen messages**setVideo(videoId, currentTime)**

This message is intercepted by the server. The server sets a confirmation timer for the bounce session and if it does not receive a nowPlaying message from the screen containing the videoId it will send a setPlaylist message to the screen.

next()

This method is used as a fallback, if the remote does not know the current playlist. Otherwise, it would use setVideo.

prev()

This method is used as a fallback, if the remote does not know the current playlist.

See e.g. [8] at Fig. 4 and 12:50-65:

In some examples, any application of applications 130 executed by processor 132 may require data from one or more of servers, such as servers 24 shown in FIG. 1. Network module 136 is configured to transmit data/requests to and receive data/responses from one or more servers via network. Network module 136 may provide received data to processor 132 for further processing. Network module 136 may support wireless or wired communication, and includes appropriate hardware and software to provide wireless or wired communication. For example, network module 136 may include an antenna, modulators, demodulators, amplifiers, and other circuitry to effectuate communication between controlled device 118 and one or more servers associated with a network. Network module 100 may communicate with one or more servers associated with the network according to a network communication protocol, such as, for example, hypertext transfer protocol (HTTP), HTTP secured by transport layer security or secure sockets

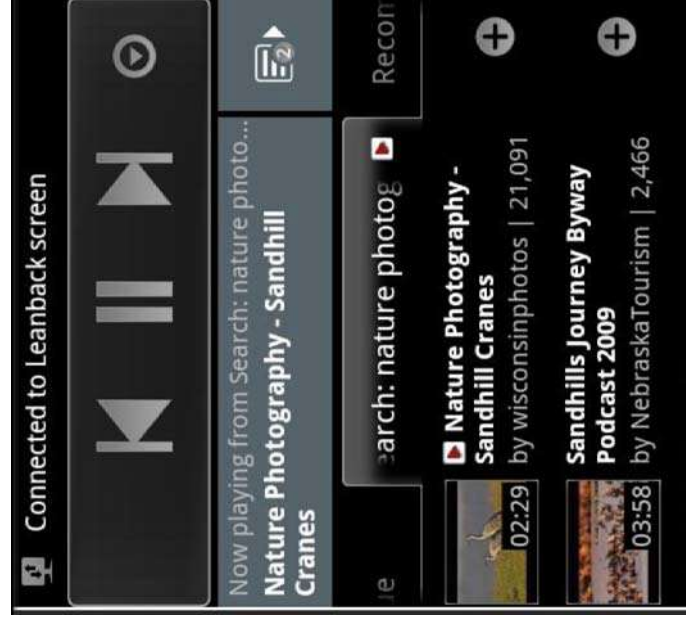
See e.g. [8] at 13:4-15:

Controlled device 118 may be used, in some examples, in conjunction with a remote control, such as remote control 14 shown in FIG. 1, remote controls 62 shown in FIG. 2, or remote control 75 shown in FIG. 3. For example, storage device 92 may store application instructions associated with a video application or web browser for displaying video content from the World Wide Web (e.g., YouTube® content, Hulu® content, Netflix® content, etc.). A user may interact with user interface 120 to execute the video or web browser

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

application. Processor 120 then executes the video or web browser application and causes display 124 to display content to the user.

See e.g. [9]:



See for example source code for the Google MDx server before ~~12.30.2011~~ 12.1.2011, including for example server source code located at google3/java/com/google/youtube/lounge, and receiver source code located at google3/flash/as3/com/google/youtube/ and google3/flash/actionscript/com/google/or google3/video/youtube/src/web/javascript/library/tv, google3/video/youtube/src/web/javascript/library/www/remote/ and servlets located at google3/video/youtube/src/python/. See also for example source code located in

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

subdirectories [YTTV^{§43}/modules/leanback](#) and [YTTV/application/YTTVF/net/browserchannel/](#), [YTTVF/youtube/](#), [google3/video/youtube/src/python/](#).

See also for example source code located in subdirectories

[google3/video/youtube/src/web/javascript/library/tv/](#)

[google3/video/youtube/src/web/javascript/library/www/](#)

[google3/java/com/google/net/browserchannel/](#), [google3/javascript/closure/net/](#)

See also e.g.: [YTTV/modules/leanback/](#)

[YTTV/application](#)

Further, on August 2, 2022, this Court construed the term “[“playback queue”](#)” as “a list of multimedia content selected for playback.” The Court indicated that a “[“playback queue”](#)” is not limited to a list of multimedia items selected by the user for playback and need not have multiple media items. At least under the Court’s construction of “[“playback queue”](#)” and Sonos’s interpretation of “[“remote”](#)” and “[“the instruction,”](#)” the YT Remote System’s shared “[“party queue”](#)” is a “[“remote playback queue”](#)” and the YouTube Remote system satisfies this limitation.

For example, the Lounge servers receives the SET PARTY PLAYLIST message, which causes the Lounge server to call [sendPartyQueueToScreen](#). This function causes the Lounge servers (cloud-based computing system) to send a SET PLAYLIST and a SET VIDEO message to the one or more selected playback devices (YouTube screens). The SET PLAYLIST message contains the party queue (and this process is repeated with the UPDATE PLAYLIST message each time the party queue is modified by a host or guest user). Thus, at least under Sonos’s interpretation, the YouTube Remote system satisfies this limitation by sending a SET PLAYLIST message or UPDATE PLAYLIST message that configures the playback device to communicate with the Lounge server in order to obtain data identifying a next one or more media items that are in the party queue (e.g., the videoIds in the party queue) and uses the obtained data to retrieve at least one media item in the remote playback queue from Google’s Bandaid CDN servers which is then played back

^{§43} Throughout, “YTTV” refers to

[YTTVLeanback12022011/google3/flash/as3/com/google/youtube](#) [YTTVLeanback1202011/google3/flash/as3/com/google/youtube/](#)

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Alternatively, a SET_PLAYLIST or UPDATE_PLAYLIST message configures the playback device to communicate with the YouTube back-end servers (e.g., servers associated with the Player service) in order to obtain data identifying a next one or more media items that are in the remote playback queue (e.g., streaming/bandaidd URLs and video data), and uses the obtained data to retrieve at least one media item in the remote playback queue from the cloud-based media service (Google's Bandaidd CDN) that is then played back.

Further, while the YouTube Remote system's SET_PLAYLIST message includes within it the videoids for the party queue, it would have also been at least obvious for the SET_PLAYLIST message to not include the videoids in the party queue and instead trigger the playback device to retrieve those videoids in a separate message (rather than as part of the same message). The remainder of the flow would remain the same, with the videoids being used to retrieve at least one media item in the remote playback queue from the Bandaidd CDN and play back the retrieved at least one media item.

See, e.g., [12]:

Party Mode

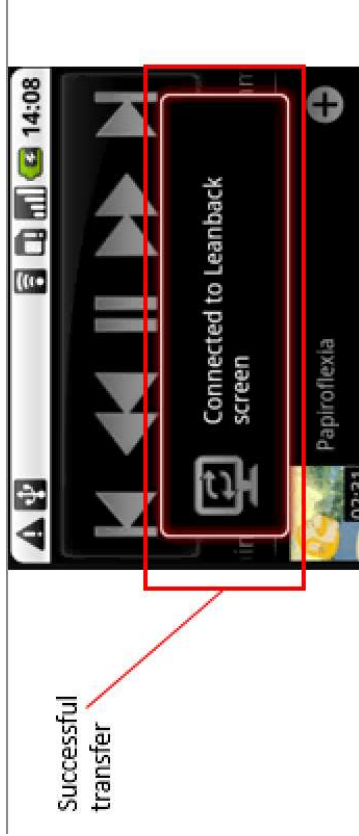
Party mode allows multiple users to connect to the same screen, queue up and watch videos together. All members of the party can add/reorder/delete videos to the same 'Shared Queue' of videos, which is being played on the Leanback screen. It basically solves this problem: <http://xkcd.com/920/>.

I've designed and implemented the shared queue management(both android app and server) cl/18357652 cl/18343670

Redesigned the party mode initialization to be invitation based: cl/18677970, cl/18636464

Launched in the version 2 of the remote.

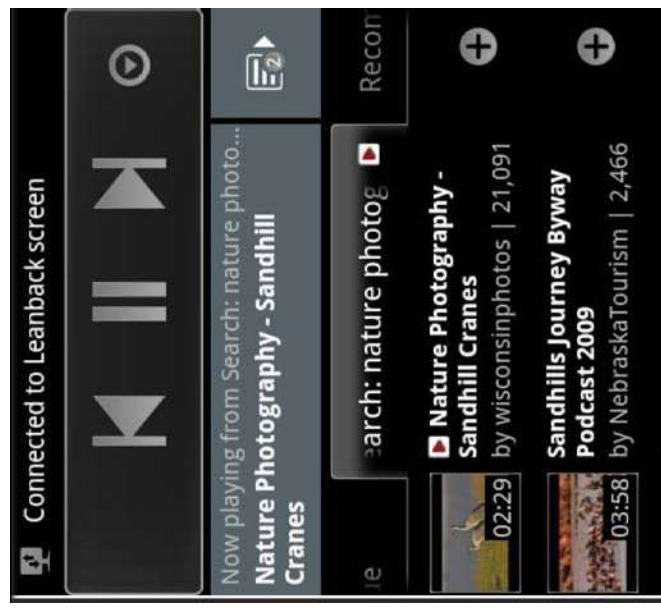
To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Riders I-J. Further discussion of the obviousness of this claim element is provided in Google's Invalidity Contentions Cover Pleading.

<p>[1f]</p>	<p>detecting an indication that playback responsibility for the remote playback queue has been successfully transferred from the computing device to the at least one given playback device; and</p>	<p>The YT Remote System discloses detecting an indication that playback responsibility for the remote playback queue has been successfully transferred from the computing device to the at least one given playback device.</p> <p>See e.g. [1]:</p> <div data-bbox="365 493 722 1323"><p>Successful transfer</p></div> <p>See e.g. [8] at 5:29-41:</p> <p>In the example shown in FIG. 1, remote control 14 includes a user interface 26 that may be used to present information to a user. For example, user interface 26 may display controls 30 and information 34 associated with content being played on controlled device 18. Controls 30 may depend on the capability of remote control 14 or controlled device 18, and include, for example, fast forward, reverse, skip ahead or back, play, stop, move to new content, etc. The type and quantity of information 34 may also depend on the capability of remote control 14 and controlled device 18, and include, for example, playback information Such as time remaining of content, playlist information, content rating information, etc.).</p> <p>See [8] e.g. at 5:42-63:</p> <p>Controlled device 18 may include a variety of network enabled devices, such as a network enabled television, set top box, personal video recorder, or other device capable of being network-connected and controlled remotely. In an example, controlled device 18 is an Internet-connected television that is configured to receive signals from and transmit signals to network 14. For example, controlled device 18 may be configured to initiate contact with servers 24. For example, controlled device 18 may notify servers 24 that controlled device 18 is connected to network 22. Controlled device 18 may notify servers 24, for example,</p>
-------------	--	---

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

automatically upon being powered on. In another example, a user may log in to a user account maintained by the servers 24 using controlled device 18, thereby notifying servers 24 that controlled device 18 is connected to network 22. Controlled device 18 can also be configured to transmit a message to servers 24 of network 22 that identifies controlled device 18, which can be used by servers 24 to pair controlled device 18 with remote control 14. The message may also contain notification or content data for updating a user interface of remote control (e.g., indicating completion of a task, such as completing playback of content).

See e.g. [9]:



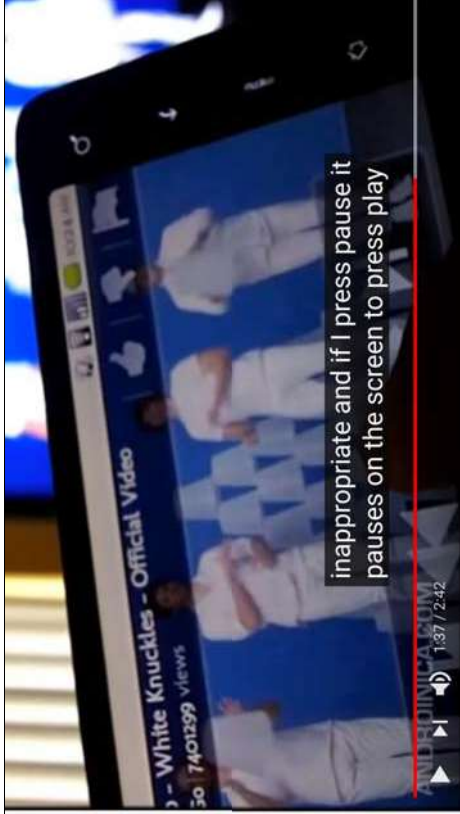
See for example source code for the Android application before ~~12.30.2011~~ 12.1.2011, including for example source code located at YTR/src/com/google/android/ytremote/, and YTR/res/. See also for example source code located in subdirectories YTR/res/values and YTR/src/com/google/android/ytremote, and YTVF/youtube/, YTVF/net/browserchannel/.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p><u>See also for example source code located in subdirectories google3/video/youtube/src/web/javascript/library/tv/.</u></p> <p><u>See also e.g.: YTR/res/values; YTR/src/com/google/android/ytremote</u></p> <p><u>Further, on August 2, 2022, this Court construed the term “playback queue” as “a list of multimedia content selected for playback.” The Court indicated that a “playback queue” is not limited to a list of multimedia items selected by the user for playback and need not have multiple media items. At least under the Court’s construction of “playback queue” and Sonos’s interpretation of “remote,” the YT Remote System’s shared “party queue” is a “remote playback queue.” Thus, under the Court’s construction and Sonos’s interpretation, a YouTube Remote application with party mode discloses and renders obvious detecting an indication that playback responsibility for the remote playback queue has been successfully transferred from the computing device to the at least one given playback device and stops local playback and updates the UI. See Bhattacharjee Decl., ¶¶134, 143 158 (discussing “[loungeScreenConnected]” and “Screen_Connected” messages and stopping of video).</u>⁴⁴</p> <p>To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Riders I-J. Further discussion of the obviousness of this claim element is provided in Google’s Invalidity Contentions Cover Pleading.</p>
[1g]	<p>after detecting the indication, transitioning from i) the first mode in which the computing device is configured for playback of the remote playback queue to ii) a second mode in which the computing device is configured to control the at least one given playback device’s playback of the remote playback queue and the computing device is no longer</p> <p>The YT Remote System discloses after detecting the indication, transitioning from i) the first mode in which the computing device is configured for playback of the remote playback queue to ii) a second mode in which the computing device is configured to control the at least one given playback device’s playback of the remote playback queue and the computing device is no longer configured for playback of the remote playback queue.</p> <p>See e.g. [4] the user’s phone (the control device) allows the user to press pause or play on the phone and the leanback screen in turns pauses or plays the video:</p>

⁴⁴ See also, e.g., [ytremote\WatchActivity.java at 205-219, 1158-1196](#).

configured for playback of the remote playback queue.



See e.g. [5]

YouTube Remote is a simple but effective remote tool for controlling YouTube Leanback from the comfort of your Android device. One of the best features of YouTube Remote is that it isn't just a remote control device for YouTube Leanback, it's also a compact YouTube viewer.

You can, for example, preview a video on your Android device before kicking it over to the playlist for your monitor or television. Once you've queued up a video to play on the big screen you can then turn off the remote function and continue to preview and add more videos to the queue.

See e.g. [8] at 5:29-41:

In the example shown in FIG. 1, remote control 14 includes a user interface 26 that may be used to present information to a user. For example, user interface 26 may display controls 30 and information 34 associated with content being played on controlled device 18. Controls 30 may depend on the capability of remote control 14 or controlled device 18, and include, for example, fast forward, reverse, skip ahead or back, play, stop, move to new content, etc. The type and quantity of information 34 may also depend on the capability of

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

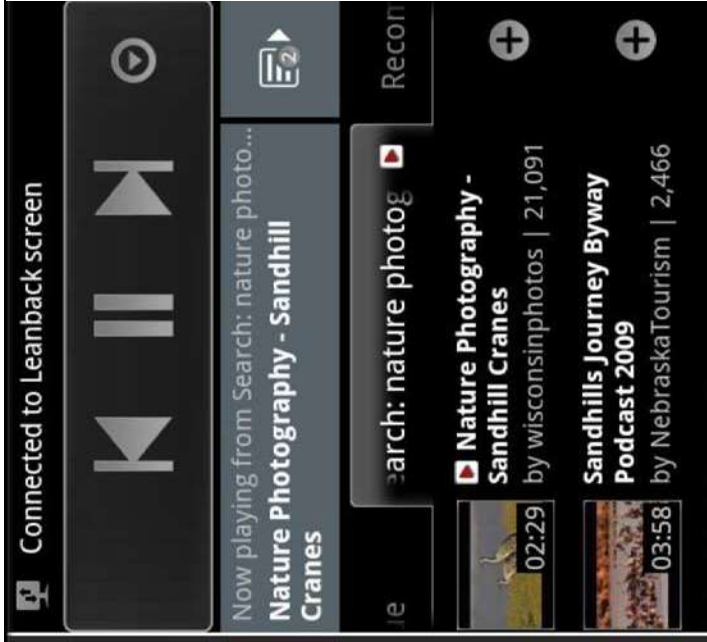
remote control 14 and controlled device 18, and include, for example, playback information Such as time remaining of content, playlist information, content rating information, etc.).

See e.g. [8] at 18:55-19:22:

FIG. 9 is a flowchart illustrating an example operation of a controlled device communicating with a network server, in accordance with one aspect of the present disclosure. For purposes of illustration only, the method of FIG. 9 is described with respect to networked environment 10 of FIG. 1, though various other systems and/or devices may be utilized to implement or perform the method shown in FIG. 9. In some examples, server 24 receives a message from controlled device 18 having a controlled device identifier and content information (250). For example, the message from controlled device 18 may contain an SID issued by servers 24 that identifies controlled device 18 as being part of a session. In addition, the message may contain content information intended to notify a user of an event regarding controlled device 18, or to prompt a user of remote control 14 to take an action (e.g., notification that playback has stopped, notification that playback of new content has begun, and the like). The content information may be used, for example, to update a user interface of remote control 14. After receiving the message from controlled device 18, server 24 retrieves a remote control identifier that identifies one or more remote controls 14 intended to receive the content information (224). For example, server 24 may query a database of stored identification numbers to determine which remote control 14 is associated with the session that includes the remote control identifier. Server 24 then transmits a message to the intended recipients (one or more remote controls 14) of the content information (258). In Some examples, server 24 forwards the content information from the first message directly to one or more remote controls 14. In other examples, server 24 may process and/or repackage the content information of the message from controlled device 18 into a new message, which can be sent to the intended recipients of the content information.

See e.g. [9]:

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

		 <p>The screenshot shows a mobile application interface. At the top, it says "Connected to Leanback screen". Below this is a music player with a play button, a progress bar, and a title "Now playing from Search: nature photo... Nature Photography - Sandhill Cranes". Below the player is a "Recommendations" section with two items: "Nature Photography - Sandhill Cranes" by wisconsinphotos (21,091) and "Sandhills Journey Byway Podcast 2009" by NebraskaTourism (2,466). Each item has a plus icon next to it.</p>
	<p>See e.g. [10]:</p> <p>“The system even works when the user logs into multiple Leanback browsers; remote control operations are seamlessly sent to all browsers.”</p> <p>See e.g. [11]</p>	

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

Update: Once we got everything rolling, we were able to get a better impression of the app. While it was a bit slow to open on our Galaxy S phone, once it is up, it worked smoothly, scrolling side to side through various queues of types of content and our favorites list. While the task of pulling up Leanback in a browser window or even on a Google TV device makes it ill-suited for viewing just one video at a time, where it excels is building a up a queue of videos and sending them over all at once. It will work on multiple screens at the same time as well, but there's no AirPlay-style syncing to be had, if one of them starts to slow down or buffer it will simply continue lagging behind, and without any volume controls or ability to reach other functions, you'll still need to keep other remotes handy.

See for example Google source code referenced in elements [1pre]-[1f] above. See also for example source code located in subdirectories YTR/src/com/google/android/ytremote, and

google3/video/youtube/src/web/javascript/library/tv/,
google3/java/com/google/net/browserchannel/, google3/javascript/closure/net/

See also e.g.: YTR/src/com/google/android/ytremote/adaptar:

YTR/src/com/google/android/ytremote:

YTR/src/com/google/android/ytremote/backend/browserchannel:

YTR/src/com/google/android/ytremote/backend

Further, on August 2, 2022, this Court construed the term “playback queue” as “a list of multimedia content selected for playback.” The Court indicated that a “playback queue” is not limited to a list of multimedia items selected by the user for playback and need not have multiple media items. At least under the Court’s construction of “playback queue” and Sonos’s interpretation of “remote,” the YT Remote System’s shared “party queue” is a “remote playback queue.” Thus, under the Court’s construction and Sonos’s interpretation, a YouTube Remote application with party mode discloses or renders obvious detecting the indication, transitioning from i) the first mode in which the computing device is configured for playback of the remote playback queue to ii) a second mode in which the computing device is configured to control the at least one given playback device’s playback of the remote playback queue (the YouTube Remote application can control playback of the party playlist on the YouTube screens) and the computing device is no longer configured for playback of the remote playback queue (playback on the YouTube Remote application

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

		stops). See Bhattacharjee Decl., ¶¶ 134, 143, 158 (discussing “loungeScreenConnected” and “Screen Connected” messages and stopping of video). ⁴⁵
		To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Riders I-J. Further discussion of the obviousness of this claim element is provided in Google’s Invalidity Contentions Cover Pleading.
[2]	The computing device of claim 1, wherein the instruction comprises an instruction for the cloud-based computing system associated with the media service to provide the data identifying the next one or more media items to the given playback device for use in retrieving the at least one media item from the cloud-based computing system associated with the cloud-based media service.	The disclosures in independent claim [1] are hereby incorporated by reference. In addition, YT Remote System includes the instruction comprising an instruction for the cloud-based computing system associated with the media service to provide the data identifying the next one or more media items to the given playback device for use in retrieving the at least one media item from the cloud-based computing system associated with the cloud-based media service.

⁴⁵ See also, e.g., [ytremote\WatchActivity.java at 205-219, 1158-1196](#).

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>See e.g. [7] Remote to Server, Screen to Server:</p> <p>Remote to Server messages</p> <p>setPlaylist(videoids, videoid, currentTime)</p> <p>videoids - comma separated video id values representing the current playlist currentTime - playback position in the video in seconds videoid - id of the video currently playing, must be part of the playlist</p> <p>The remote informs the server what its current playlist is Sent when the remote connects to a screen If there is no playlist in the session, the playlist sent by the remote will become the current playlist. If there already is one, the playlist will be ignored. The server will also send a request for nowPlaying to the screen</p> <p>addVideo(videoid) insertVideo(videoid) addVideos(videoids) moveVideo(videoid, delta) removeVideo(videoid) clearPlaylist()</p> <p>These messages change the current playlist on the server. If a change occurs, the server will send updates to the remotes (via playlistModified) and to the screen (via updatePlaylist)</p>
--	--

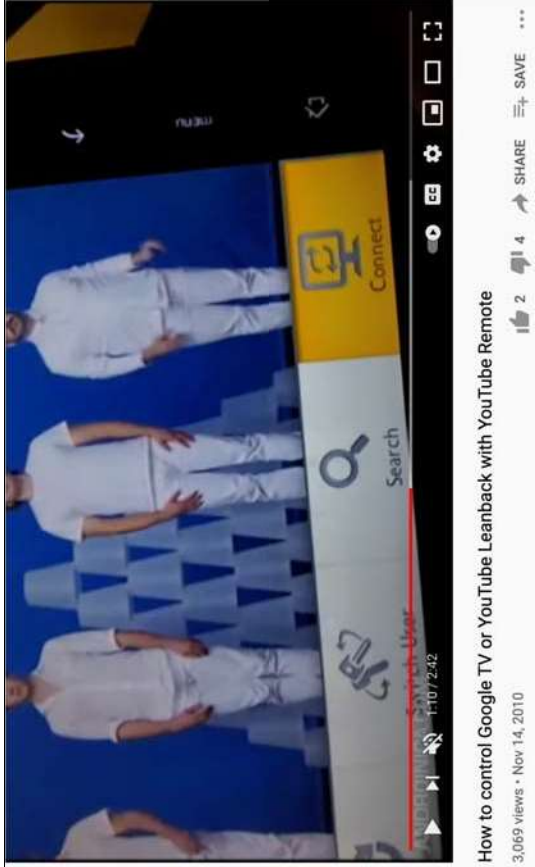
	<p>Screen to Server messages</p> <p>nowPlaying(video_id, current_time) video_id - the encrypted video id of the currently playing video current_time - playback position in the video state - the video player state: unstarted (-1), ended (0), playing (1), paused (2), buffering (3), video cued (5)</p> <p>Screen informs the server what the currently playing video is. The screen sends this message after any video data change event (such as on next, on previous, or when the next video auto-plays).</p> <p>The server will forward this message to all remotes in the session. If the video_id is not found in the current server playlist, the server will issue a getPlaylist() message to the screen.</p> <p>onStateChange(state) state - the new video player state: unstarted (-1), ended (0), playing (1), paused (2), buffering (3), video cued (5)</p> <p>Implemented via the our player's JS API, when we receive a onStateChange event we relay that information to the remotes.</p> <p>confirmPlaylistUpdate(updated) updated - true/false: whether the playlist was updated</p> <p>As a response to updatePlaylist, confirms whether the playlist was updated or not: it could be rejected if the playing video is not part of the playlist being sent.</p> <p><i>See e.g. [8] at 17:21-43:</i></p> <p>In some examples, server 24 receives a message from remote control 14 having a remote control identifier and control information (220). For example, the message from remote control 14 may contain an SID issued by servers 24 that identifies remote control 14 as being part of a session. In addition, the message may contain control information intended to alter the operation of one or more controlled devices 18 (e.g., stop playback, begin next payback item, etc.). After receiving the message from remote control 14, server 24 retrieves a controlled device identifier that identifies one or more controlled devices 18 intended to receive the control information (224). For example, server 24 may query a database of stored identification numbers to determine which controlled device is associated with the session that includes the remote control identifier. Server 24 then transmits a message to the intended recipients (one or more controlled devices 18) of the control information (228). In some examples, server 24 forward the control information from the first message directly to one or more controlled devices 18. In other examples, server 24 may process and/or</p>	
--	---	--

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>repackage the control information of the message from remote control 14 into a new message, which can be sent to the intended recipients of the control information.</p> <p><u>See for example Google source code referenced in elements [1pre]-[1g] above. See also for example source code located in subdirectories YTR/src/com/google/android/ytreremote/, YTL/browserchannel, YTL/model and YTTV/modules/leanback, YTTVF/youtube/, google3/video/youtube/src/python/, YTTVF/net/browserchannel/, and google3/video/youtube/src/web/javascript/library/tv/, google3/video/youtube/src/web/javascript/library/www/, google3/java/com/google/net/browserchannel/, google3/javascript/closure/net/</u></p> <p><u>See also e.g.: YTR/src/com/google/android/ytreremote/backend/model: YTR/src/com/google/android/ytreremote; YTL/browserchannel; YTL/model; YTTV/modules/leanback</u></p> <p><u>Further, at least under the Court’s construction of “playback queue” and Sonos’s interpretation of “remote,” the YT Remote System’s shared “party queue” is a “remote playback queue.” Thus, under the Court’s construction and Sonos’s interpretation, a YouTube Remote application with party mode performs this limitation for the same reasons discussed in Limitation 1[d]-[1e].</u></p> <p>To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Riders I-J. Further discussion of the obviousness of this claim element is provided in Google’s Invalidity Contentions Cover Pleading.</p>
[4a]	<p>The computing device of claim 1, wherein the representation of the one or more playback devices comprises at least one selectable indicator for a group of playback devices that includes the given playback device and one or more other playback devices that are to be configured for</p> <p>The disclosures in independent claim [1] are hereby incorporated by reference. In addition, YT Remote System includes the representation of the one or more playback devices comprises at least one selectable indicator for a group of playback devices that includes the given playback device and one or more other playback devices that are to be configured for synchronous playback of the remote playback queue.</p> <p><i>See e.g. [4]:</i></p>

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

synchronous playback of the remote
playback queue, and



See e.g. [8] at 17:44-48:

Server 24 may, in some examples, initialize a group session upon receiving the message from remote control 14. For example, server 24 may maintain a session that includes the identifiers for all of the components sending and receiving messages (e.g., remote control(s) 14 and controlled device(s) 18).

See e.g. [9]:

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>See e.g. [10]:</p> <p>“The system even works when the user logs into multiple Leanback browsers; remote control operations are seamlessly sent to all browsers.”</p> <p>See e.g. [11]:</p>	

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

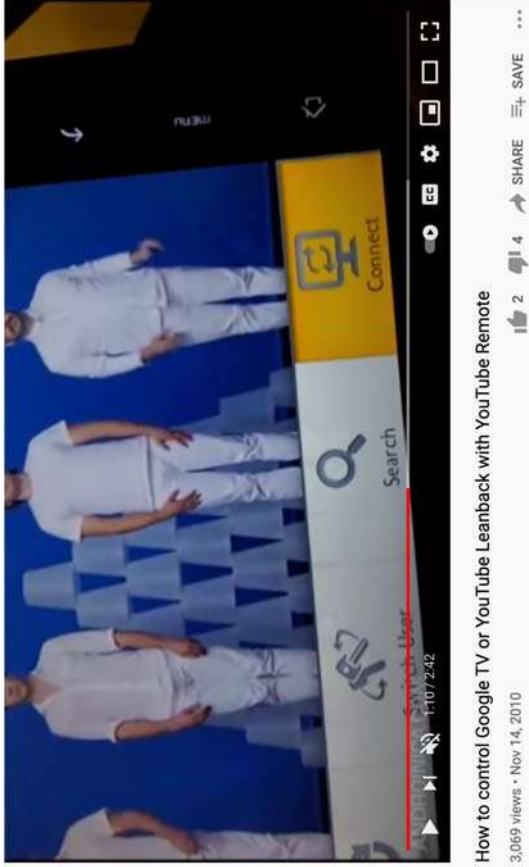
	<p>Update: Once we got everything rolling, we were able to get a better impression of the app. While it was a bit slow to open on our Galaxy S phone, once it is up, it worked smoothly, scrolling side to side through various queues of types of content and our favorites list. While the task of pulling up Leanback in a browser window or even on a Google TV device makes it ill-suited for viewing just one video at a time, where it excels is building a up a queue of videos and sending them over all at once. It will work on multiple screens at the same time as well, but there's no <u>Airplay</u>-style syncing to be had, if one of them starts to slow down or buffer it will simply continue lagging behind, and without any volume controls or ability to reach other functions, you'll still need to keep other remotes handy.</p> <p><u>See for example Google source code referenced in elements [1pre]-[1f] above. See also for example source code located in subdirectories YTR/src/com/google/android/ytreremote/, YTL/browserchannel and YTL/model and google3/video/youtube/src/web/javascript/library/www/remote/</u></p> <p><u>See also e.g.: YTR/src/com/google/android/ytreremote/adapt: YTR/src/com/google/android/ytreremote; YTL/browserchannel.; YTL/model To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Riders J-K. Further discussion of the obviousness of this claim element is provided in Google's Invalidity Contentions Cover Pleading. YTL/model To the extent it is argued that the "connect" button is not is not the claimed "representation of one or more playback devices in a media playback system" and that a "device-picker" is required, this limitation is anticipated if Sonos is not entitled to its July 15, 2011 invention date because Google implemented the device-picker in its December 30, 2011 source code and released it no later than January of 2012 in Version 3 of the YouTube Remote. Bhattacharjee Decl., ¶170. To the extent Sonos is entitled to its July 15, 2011 invention date, this limitation is at least obvious in view of one or more of the YouTube Remote Patent's disclosure of a device-picker, Google's Tungsten/Nexus Q device with device-picker, Apple Airplay, Sonos's own prior art, and/or the Al-Shayk patent. Bhattacharjee Decl., ¶¶ 27-34, 165-174. Indeed, the Court's August 2, 2022 Order concluded that it would have been obvious to add the device-picker to the YouTube Remote application. Dkt. No. 316 at 14-17.</u></p>
[4b]	<p>wherein the user input indicating the selection of at least one given</p> <p>See element [4a] above.</p>

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>playback device from the one or more playback devices comprises user input indicating a selection of the group of playback devices.</p>	<p>See also for example source code located in subdirectories</p> <p>YTR/src/com/google/android/ytremote/, YTR/src/com/google/android/apps/ytlounge/res/, YTR/src/com/google/android/youtube/ui, and YTR/src/com/google/android/ytremote/backend</p> <p>See also e.g.:</p> <p>YTR/src/com/google/android/ytremote/adapters;</p> <p>YTR/src/com/google/android/ytremote;</p> <p>YTR/src/com/google/android/apps/ytlounge/res/drawable-hdpi;</p> <p>YTR/src/com/google/android/apps/ytlounge/res/drawable-mdpi;</p> <p>YTR/src/com/google/android/apps/ytlounge/res/drawable;</p> <p>YTR/src/com/google/android/youtube/ui;</p> <p>YTR/src/com/google/android/ytremote/backend/browserchannel;</p> <p>YTR/src/com/google/android/ytremote/backend.</p>
[9]	<p>The computing device of claim 8, wherein the transport control operation comprises one of a play operation, a pause operation, a skip forward operation, or a skip back operation.</p>	<p>The disclosures in the independent claim are hereby incorporated by reference. In addition, YT Remote System discloses the transport control operation comprises one of a play operation, a pause operation, a skip forward operation, or a skip back operation.</p> <p>See e.g. claim element [1.g] above.</p>
[11]	<p>The computing device of claim 1, wherein displaying the representation of the one or more playback devices comprises: displaying the representation of the one or more playback devices in response to receiving a selection of a</p>	<p>The disclosures in independent claim [1] are hereby incorporated by reference. In addition, YT Remote System includes displaying the representation of the one or more playback devices comprises: displaying the representation of the one or more playback devices in response to receiving a selection of a displayed icon indicating that playback responsibility for the remote playback queue can be transferred.</p>

displayed icon indicating that playback responsibility for the remote playback queue can be transferred.

See e.g. [4]:



See e.g. [8] at 4:4-20:

According to some examples, the network service may assign each remote control and each controlled device a unique identifier. When pairing devices, the network service may utilize the unique identifier associated with each device to route communication signals properly. For example, the network service may initiate a session that includes each unique identifier of remote controls and controlled devices that are authorized to communicate with each other. The network service can then route messages to members of the session. Any number of remote controls may be paired with a single controlled device and one remote control may be paired to any number of controlled devices. When pairing multiple remote controls and multiple controlled devices associated with a single user, the user may identify a Subset of the remote controls as paired to a subset of the controlled devices, and manage which remote controls control which controlled devices.

See for example Google source code referenced in elements [1pre]-[11f] above.

See also for example source code located in subdirectories

[YTR/src/com/google/android/ytremote/./YTR/res/drawable-mdpi, YTR/res/menu, and YTR/src/com/google/android/ytremote/factory, and](https://source.google.com/archive/ytrete/ytremote/./YTR/res/drawable-mdpi/YTR/res/menu, and YTR/src/com/google/android/ytremote/factory, and)

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	<p>google3/video/youtube/src/web/javascript/library/tv/google3/video/youtube/src/web/javascript/library/www/.</p> <p><u>See also e.g.: YTR/src/com/google/android/ytremote/adapters/YTR/src/com/google/android/ytremote; YTR/src/com/google/android/ytremote; YTR/res/drawable-mdpi; YTR/res/menu; YTR/src/com/google/android/ytremote/factory; YTR/src/com/google/android/ytremote.</u></p> <p>To the extent it is argued that these references do not disclose this claim element, it would have at least been obvious to combine these references with the references cited in Rider J. Further discussion of the obviousness of this claim element is provided in Google's Invalidity Contentions Cover Pleading. To the extent it is argued that the "connect" button does not satisfy the claims and that a "device-picker" is required, this limitation is anticipated if Sonos is not entitled to its July 15, 2011 invention date because Google implemented the device-picker in its December 30, 2011 source code and released it no later than January of 2012 in Version 3 of the YouTube Remote. Bhattacharjee Decl., ¶170. To the extent Sonos is entitled to its July 15, 2011 invention date, this limitation is at least obvious in view of one or more of the YouTube Remote Patent's disclosure of a device-picker, Google's Tungsten/Nexus Q device with device-picker, Apple Airplay, Sonos's own prior art, and/or the Al-Shayk patent. Bhattacharjee Decl., ¶¶ 27-34, 165-174. Indeed, the Court's August 2, 2022 Order concluded that it would have been obvious to add the device-picker to the YouTube Remote application. Dkt. No. 316 at 14-17.</p>
[12pre]	A non-transitory computer-readable medium having stored thereon program instructions that, when executed by at least one processor, cause a computing device to perform functions comprising:
[12a]	operating in a first mode in which the computing device is configured for playback of a remote playback queue
	The disclosures in independent claim [1] are hereby incorporated by reference. <i>See e.g.</i> claim element [1pre] above.
	The disclosures in independent claim [1] are hereby incorporated by reference. <i>See e.g.</i> claim element [1a] above.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	provided by a cloud-based computing system associated with a cloud-based media service;	
[12b]	while operating in the first mode, displaying a representation of one or more playback devices in a media playback system that are each i) communicatively coupled to the computing device over a data network and ii) available to accept playback responsibility for the remote playback queue;	The disclosures in independent claim [1] are hereby incorporated by reference. <i>See e.g.</i> claim element [1b] above.
[12c]	while displaying the representation of the one or more playback devices, receiving user input indicating a selection of at least one given playback device from the one or more playback devices;	The disclosures in independent claim [1] are hereby incorporated by reference. <i>See e.g.</i> claim element [1c] above.
[12d]	based on receiving the user input, transmitting an instruction for the at least one given playback device to take over responsibility for playback of the remote playback queue from the computing device, wherein the instruction configures the at least one given playback device to (i) communicate with the cloud-based computing system in order to obtain data identifying a next one or more media items that are in the remote playback queue, (ii) use the obtained data to retrieve at least one media item in the remote playback queue from the cloud-based media service; and (iii)	The disclosures in independent claim [1] are hereby incorporated by reference. <i>See e.g.</i> claim element [1d] and [1e] above.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

	play back the retrieved at least one media item;	
[12e]	detecting an indication that playback responsibility for the remote playback queue has been successfully transferred from the computing device to the at least one given playback device; and	The disclosures in independent claim [1] are hereby incorporated by reference. <i>See e.g.</i> claim element [1f] above.
[12f]	after detecting the indication, transitioning from i) the first mode in which the computing device is configured for playback of the remote playback queue to ii) a second mode in which the computing device is configured to control the at least one given playback device's playback of the remote playback queue and the computing device is no longer configured for playback of the remote playback queue.	The disclosures in independent claim [1] are hereby incorporated by reference. <i>See e.g.</i> claim element [1g] above.
[13]	The non-transitory computer-readable medium of claim 12, wherein the instruction comprises an instruction for the cloud-based computing system associated with the cloud-based media service to provide the data identifying the next one or more media items to the given playback device for use in obtaining the at least one media item from the cloud-based computing system associated with the cloud-based media service.	<i>See</i> claim [2] above.

HIGHLY CONFIDENTIAL – ATTORNEY EYES ONLY

[16]	<p>The computing device of claim 1, further comprising program instructions stored on the non-transitory computer-readable medium that, when executed by the at least one processor, cause the computing device to perform functions comprising:</p> <p>before displaying the representation of the one or more playback devices, receiving an indication that the one or more playback devices in the media playback system are available to accept playback responsibility for the remote playback queue.</p>	<p>YT Remote System discloses the computing device of claim 1 further comprising program instructions stored on the non-transitory computer-readable medium that, when executed by the at least one processor, cause the computing device to perform functions comprising before displaying the representation of the one or more playback devices, receiving an indication that the one or more playback devices in the media playback system are available to accept playback responsibility for the remote playback queue. <i>See e.g.</i> claim element [1b] above.</p>
------	---	---